Mesh-based Tools to Analyze Deep Reinforcement Learning Policies for Underactuated Biped Locomotion

Nihar Talele and Katie Byl

Abstract— In this paper, we present a mesh-based approach to analyze stability and robustness of the policies obtained via deep reinforcement learning for various biped gaits of a five-link planar model. Intuitively, one would expect that including perturbations and/or other types of noise during training would likely result in more robustness of the resulting control policy. However, one would like to have a quantitative and computationally-efficient means of evaluating the degree to which this might be so. Rather than relying on Monte Carlo simulations, our goal is to provide more sophisticated tools to assess robustness properties of such policies. Our work is motivated by the twin hypotheses that contraction of dynamics, when achievable, can simplify control and that control policies obtained via deep learning may therefore exhibit tendency to contract to lower-dimensional manifolds within the full state space, as a result. The tractability of our mesh-based tools in this work provides some evidence that this may be so.

I. INTRODUCTION

Although legged locomotion is less energy efficient than wheeled locomotion on relatively mild terrain, it offers the advantage of more flexibility in the variety of terrain that can be traversed. However, this flexibility often comes with added complexity of control, particularly as legged locomotion can involve phases of underactuation. Two approaches have become prominent in recent times for the control of legged locomotion. Model-based trajectory optimization has shown impressive results, for example in its application within the DARPA Robotics Challenge (DRC) [1], [2]. Also, with the advent of improved computational capabilities, the field of deep reinforcement learning (DRL) is now being successfully applied to generate control policies for complicated dynamical systems like humanoids [3].

Model-based trajectory optimization methods, as described in [4], [5], involve the generation of a trajectory for the system that optimizes a certain cost function, such as energy minimization. This optimal solution is then used as a reference trajectory for the actual system through the use of a low level controller. The low level controller enforces contraction of the system on a low dimensional manifold; i.e., for the underactuated dynamics, some subset of directly-controlled degrees of freedom (DOFs) converge rapidly to desired trajectories, compared to other (passive, slower, but still stable) DOF(s). This contraction onto lower dimensional manifolds can often allow us to avoid the classic of "curse of dimensionality" in implementing discrete approximations to analyze the resulting nonlinear dynamics; we exploit this

contraction to apply meshing tools to evaluate the performanc of control policies for such systems. In our own group, we have used model-based trajectory optimizations to corroborate human data studies with model-based energy-optimal gaits [6] and to explore non-intuitive motions for underactuated rolling locomotion systems [7]. While relationships between the cost function, constraints in the optimization, low level controller, and the robustness of the obtained policy often have some intuitive characteristics, it is not so clear what quantitative effects the choice of reward function has on the robustness of the obtained policy. With this goal in mind, and with application to quantification of deep learning policies as a particular aim, we analyze locomotion control polices obtained via deep reinforcement learning and propose the use of meshing tools to quantify stability and robustness in terms of failure rates.

In [8], Byl and Tedrake propose the mean first passage time (MFPT) as a metric to evaluate the robustness of a rimless wheel and of the compass gait walker, each on rough terrain. In this framework, robust walking is viewed as a metastable dynamic process: capable of very long periods in near-limit-cycle behavior but ultimately guaranteed to eventually fall. Saglam and Byl build upon this work by developing a reachability-based approach to non-uniform meshing [9], [10]. In all of these works, the only noise considered is variability in terrain height, i.e., walking on rough terrain. In [11], Talele and Byl demonstrate that these mesh-based tools can also be extended to additional noise cases, including push disturbances throughout the gait cycle. In our current work, we expand the applicability of our meshing tools by first showing that it is possible to use the meshing analysis on policies generated via deep reinforcement learning and then use these tools to perform a systematic analysis on the generated policies to obtain a correlation between the reward functions used and the robustness of the corresponding policies.

The rest of the paper is organized as follows. In Section II, we present the 5-link model created in MuJoCo [12] that we use for our simulation as well as some information on the deep reinforcement learning policies that we generate. In Section III, we provide details on the basic meshing concepts that we use for our analyses in the results section. Section IV presents our results, where we evaluate the robustness of the policies obtained via deep reinforcement learning for different reward functions, following which we conclude in Section V by discussing and summarizing the ideas presented along with the possible future directions in which we want to extend the current framework.

Nihar Talele and Katie Byl are with the Electrical Engineering Department at the University of California, Santa Barbara CA 93106 nihar@ucsb.edu, katiebyl@ucsb.edu
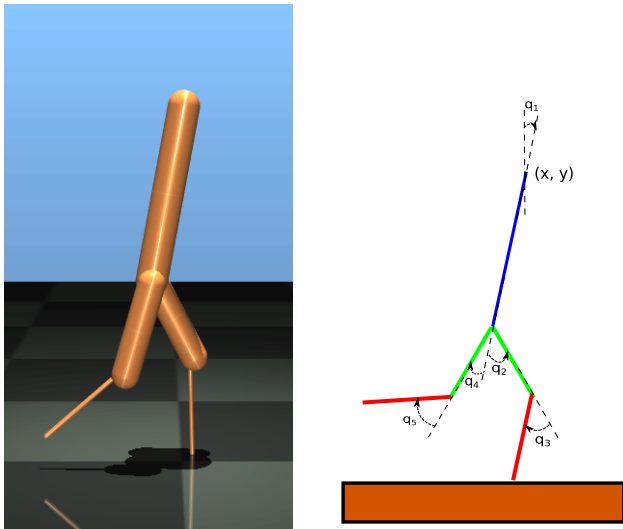
II. MODEL AND CONTROL

A. Model

Fig. 1. The 5-link biped model used in simulations. At left, the planar model in MuJoCo's simulation engine, used for our simulations. The stick figure on the right shows the degrees of freedom of the model on the left.

We use a 5-link planar biped model with point feet as shown in Fig. 1 for our analyses. All simulations for the results presented in this paper are done in MuJoCo [12]. The model has a torso, two hips and two knees. Its total mass is 70 (kg), and it stands 1.6 (m) tall when fully upright. Length, mass and inertia parameters, which are chosen to approximate a typical human, are identical to those listed for "Set 5" within [13]. The model has a total of 4 actuators (2 at hips and 2 at the knees) and 7 degrees of freedom $q := [x, \ y, \ q_1, \ q_2, \ q_3, \ q_4, \ q_5]^T$ where $[x, \ y]$ are the position coordinates of the topmost point of the torso and $[q_1, q_2, q_3, q_4, q_5]$ correspond to the five angles shown in the Fig. 1. We ignore rolling friction and set the friction model to oppose sliding in the tangential plane of two contacting bodies only. We set all the viscous damping coefficients to zero. All the contacts in the simulation are soft contacts: MuJoCo models the interaction between two bodies as a soft contact, for efficient computation. We set the integration method to Runge-Kutta (RK4) with a time step of 0.002 (s). All the torque inputs are restricted to $\pm$ 100 (N·m).

B. Control Policy

We use the Proximal Policy Optimization algorithm [14] in the openAI's baselines package [15] to train our model and obtain the control policies. The training algorithm models continuous time action space as a probability density distribution that it learns for each observation or state of the environment. While training, the algorithm samples from this distribution and then takes an action for the corresponding observation. During evaluation of a trained policy, however, instead of sampling from the learned distribution, we pick the action that has the maximum likelihood. Each control action is held at a constant value for a total of 4 consecutive time steps during simulation. Thus, even though the time step for the integration scheme we use is 0.002 (s), the action sampled for the current observation or the environment state is applied for a total of 0.008 (s). We set the maximum training episode length to 4000 time steps, and each state observation is clipped to a maximum magnitude of 10.

III. MESHING

We create our mesh $\mathbb{M}$ by deterministically mapping the reachable state space of the system on a Poincaré section for various disturbances, γ, which for this work are pushes of a variety of magnitudes and timing within the gait cycle. Because there are no constraints requiring left-right symmetry of the locomotion policy learned, we perform a Poincaré analysis on a full gait "cycle", i.e., after two steps are taken. A Poincaré section is taken when the left foot (an arbitrary choice) makes an impact with the ground. We denote the post-impact state as $s = x^+$. The mesh has one self-absorbing state (state #1) to which all failure events transition. In general, our meshing tools can allow us to calculate the optimal (switching) policy from a set of controllers, but for the current work, we simply analyze the policies individually obtained via DRL framework as explained in Section II. Once the left foot impacts the ground, we compare the post-impact state to previously observed post-impact states using the following metric:

$$d(s_i) = \min_{s_j \in \mathbb{M}} \sqrt{\sum_{k=1}^{n} (s_i(k) - s_j(k))^2}. \tag{1}$$

If $d(s_i) > d_{tr}$, where d_{tr} is some distance threshold, the state s_i is added to the mesh. A deterministic state transition matrix $T_{det}(s, \xi, \gamma)$ which describes all state transitions is maintained and updated at every iteration, where ξ is the particular DRL policy being analyzed. In case of failure, the corresponding transition goes to state #1, indicating that under the current control policy, the state has transitioned to the absorbing failure state in the mesh.

Our disturbance set consists of various pushes γ happening at different times in forward as well as backward directions. Each push occurs at the center of mass (COM) of the torso link. The disturbance profile is characterized by a certain probability distribution that we chose, and it is denoted by $P(\gamma)$. Algorithm 1 outlines our meshing procedure.

Once we have the deterministic state transition matrix, we proceed to calculate the stochastic state transition matrix and other important elements needed to perform our analyses as explained in the following sections.

A. Stochastic State Transition matrix

To analyze the performance of a particular DRL-based control policy, ξ, when subject to some disturbance profile, $P(\gamma)$, we first calculate a *stochastic* transition matrix, $T(i,j)$:

$$T(i,j) := \sum_{\gamma} P(\gamma) f_j \tag{2}$$

where

$$f_j = \begin{cases} 1, & \text{if } T_{det}(i, \pi(i), \gamma) = j \\ 0, & \text{else.} \end{cases} \tag{3}$$

Algorithm 1 Meshing Algorithm

Input: Initial matrix of states S, containing two states: s_1 (failure) and s_2 (one initial, non-failure state).
 Disturbance profile D.
 Distance threshold d_{tr}.
 Set of controllers Z.
Output: Final matrix of all states S, discretely spanning the reachable state space.
 Deterministic state transition matrix $T_{det}(s, \xi, \gamma)$.

 Initialization:
 Current state index: cur = 2
 Total number of states: nstate = 2
 while cur $\leq$ nstate **do**
  xcur $\leftarrow$ S(cur)
  for all controllers ξ in Z **do**
   for all disturbances γ in D **do**
    x, flag $\leftarrow$ simulatedynamics(xcur, ξ, γ)
    if flag = 0 (Step taken successfully) **then**
     if d(x, s) > d_{tr} $\forall$ s $\in$ S **then**
      add x to S and set $T_{det}(xcur, \xi, \gamma)$ = nstate
      nstate $\leftarrow$ nstate + 1
     else
      set $T_{det}(xcur, \xi, \gamma)$ = index of s for d(xcur, s) < d_{tr}
     end if
    else
     set $T_{det}(xcur, \xi, \gamma)$ = 1 (index to failure state)
    end if
   end for
  end for
  cur $\leftarrow$ cur + 1
 end while
 return S, T_{det}

Matrix $T(i, j)$ gives the probability of transitioning from state i to state j within mesh $\mathbb{M}$. Because all states must transition to some state under the entire probability distribution of the disturbance profile, each row of the stochastic state transition matrix sums to 1.

B. Mean First Passage Time

The concept of the mean first passage time (MFPT) for metastable (i.e., "rarely falling") walking systems is explained in detail in [16] and [8]. We highlight important results here that we will be using in the rest of the paper. To analyze not only convergence rates (eigenvalues) but also probability distributions across state space (via eigenvectors), we focus on an eigenanalysis of the *transpose*, T^T, of the stochastic transition matrix, T. First, we define a metastable distribution, ϕ, as the stationary distribution in state space, conditioned on not having entered the failure state:

$$\phi_i = \lim_{n \to \infty} Pr(X(n) = x_i | X(n) \neq x_1). \tag{4}$$

The mean first passage vector m, where m_i is the mean time for the state i to go into failure, is given by

$$m_i = \begin{cases} 0, & \text{if } i = 1 \\ 1 + \sum_{j>1} T_{ij} m_j, & \text{else.} \end{cases} \tag{5}$$

Vector m can be calculated by

$$m = \begin{bmatrix} 0 \\ (I - \hat{T})^{-1} \end{bmatrix}, \tag{6}$$

where $\hat{T}$ is T with the first column and row (corresponding to the failure state, for which $m(1) = 0$) removed. This in turn lets us calculate the system wide MFPT:

$$M = \sum_i \phi_i m_i. \tag{7}$$

This process can be time consuming and for a more efficient calculation of M, we use the eigenvalues of the stochastic transition matrix. As stated, all failures (e.g., falls) are modeled via a self-absorbing state. This enforces a structure on the stochastic state transition matrix where $T(1,1) = 1$, resulting in the largest eigenvalue of T being 1. Let λ_2 be the second largest eigenvalue of T. The magnitude of λ_2 is the probability of taking a successful step, assuming initial conditions have been forgotten and the walker is in a non-failure state; i.e., that the likelihood of currently being in any state is given by the metastable distribution, ϕ. The probability of failure on the next step is then $1 - \lambda_2$, and the probability of taking only n steps is

$$Pr(X(n) = x_1, X(n-1) \neq x_1) = \lambda_2^{n-1}(1 - \lambda_2). \tag{8}$$

When the dynamics of λ_2 are much slower than all smaller-magnitude eigenvalues, i.e., when $(1 - |\lambda_2|) << (1 - |\lambda_3|)$, initial conditions will be forgotten long before an expected failure event, and a system-wide mean first passage time, M, provides a useful metric for stability:

$$M = \frac{1}{(1 - \lambda_2)}. \tag{9}$$

Finally, note that normalizing all the non-failure-state elements in the eigenvector associated with the 2nd-largest eigenvalue of T^T (the *transpose* of T) correspondingly yields the metastable distribution, ϕ, in Eq. 4.

IV. RESULTS

We perform analyses on two DRL-trained policies, obtained under different training conditions. The first policy is trained with a reward function that encourages forward velocity; there are no perturbations, and terrain is flat. At each time step, if the walker has not fallen, the reward is incremented by the forward velocity at that instant of time; the reward function also includes a penalty 1e-3 times the norm of the torques. For the second scenario, we use the same reward function, but we now introduce push disturbances while training (still on flat terrain). For both cases, we train several policies and then pick the one that has the maximum reward at the end of the training session for our analyses. Both cases also have the same coefficient of friction of 0.5

for contact between the ground and the walker model. For meshing, we ignore the x coordinate of the top of the torso, because we perform meshing for step-to-step transitions and it does not matter from what x position the model takes the step. The mesh thus contains 13 dimensional states, of which 12 are independent. (The left foot, but not necessarily the right one, must by definition be at $y = 0$ immediately post-impact, adding a constraint and removing an independent DOF on the Poincaré section.) We explore both cases and analyze the corresponding policies obtained in the next two subsections.

A. Case 1

In this scenario, we train the policy without any external perturbations, for flat ground walking. The policy with the best reward function, chosen for our analyses, results in a significantly asymmetric forward motion. As previously mentioned, we consider state-to-state transitions for a full gait cycle, defined as one complete sequence of right leg and subsequent left leg contacts with the ground. The states on the Poincaré section then represent the post impact state of the system each time the left leg makes contact with the ground. Fig. 2 illustrates a typical set of consecutive, post-impact states during locomotion (again, with no perturbations) for 250 gait cycles.
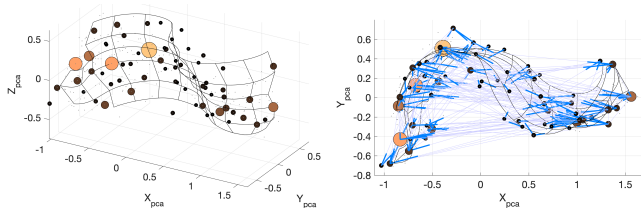

Fig. 2. A principal component analysis (PCA) is used to visualize post-impact states visited for DRL-trained flat-ground locomotion, subject to no disturbances. A non-uniform mesh of 64 points is shown, achieved by applying our meshing algorithm to a set of 250 consecutive gait cycles. Larger markers depict more frequently visited locations. Here, three principle components account for 94% of the variance of the normalized Poincaré states. At left, states fall near the depicted curvy, 2D surface within the 3D PCA space. At right, straight lines show step-to-step transitions, with terminal ends shown as thicker, blue segments. The gait has no observable limit cycle, and it appears to be both stable (never-falling) and chaotic.

As Figure 2 illustrates, applying the closed-loop DRL control policy to flat-ground walking with no noise inputs does not result in any discernable (period-n) limit cycle but instead exhibits chaotic behavior. Based on our numeric methods for mapping the reachable state space, post-impact states are bounded and seem to contract onto a lower-dimensional and bounded region of the full state space.

To examine the robustness of the given policy to push disturbances, we proceed by calculating a mesh. From simulations, we find the complete two-step gait cycle takes about 0.35 (s). Based on this, for building the mesh, we define a threshold of 0.3 (s) after which any post impact (for the left foot) state of the system, s_i, is added to the mesh if $d(s_i) > d_{tr}$, as described in Section III.

For illustrative purposes within Case 1, just four possible disturbance pushes are considered: a push of either $+1000$

(N) or -1000 (N) is applied at the COM of the torso for a duration of 0.008 (s), starting at either 0.1 or 0.25 (s) into the gait cycle. Along with these 4 disturbances, we also consider a no-push disturbance, i.e., a push with 0 magnitude. The probability distribution for these disturbances is given by

$$P(\gamma) = \begin{cases} 0.4, & \text{if no disturbance} \\ 0.6/4, & \text{else.} \end{cases} \quad (10)$$

For the given policy and disturbance profile, we obtain a mesh with 28,757 states when $d_{tr} = 0.6$. The mean first passage time for the system for the given probability distribution comes out to only about 32 steps. Once the mesh is generated, our tools allow us to efficiently calculate the MFPT for any given probability distribution. For example, if we reduce the probability of a disturbance so that

$$P(\gamma) = \begin{cases} 0.8, & \text{if no disturbance} \\ 0.2/4, & \text{else,} \end{cases} \quad (11)$$

the MFPT increases to approximately 117 steps. We also calculate a mesh for other values of d_{tr}, to analyze how varying this threshold distance changes the number of states in the mesh. If the mesh points occupy an n-dimensional subspace within the full state space, then the number of mesh points, N, required to span this subspace should grow as $N \propto d_{tr}^{-n}$, meaning a loglog plot of d_{tr} vs N would have slope $-n$. (See [11] for details.)

d_{tr}	0.6	0.7	0.8
N (# of mesh points)	28,757	14,891	8,517

TABLE I. N VERSUS d_{tr}, SHOWING $n \approx 4.23$ FOR CASE 1.

Table I shows this variation. A line fitting $x = log(d_{tr})$ vs $y = log(N)$ for these data has slope -4.2, showing that as we mesh more finely, the size of the mesh grows with dimensionality $n \approx 4.23$. Intuitively, the dimensionality of reachable state space will depend on both the sparsity of the perturbation set (recall we only include 4 non-zero perturbations here) as well as the contracting nature of the control policy. Rather than focusing on a more dense set of perturbations for Case 1, we instead focus on comparing the effects of adding perturbations during training (in Case 2).
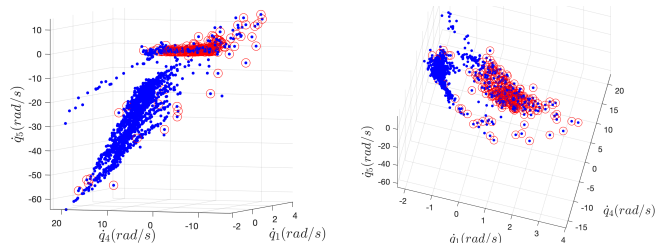

Fig. 3. A 3D section of the full 13D Poincaré mesh states for Case 1. Subplots show two viewpoints of the same data. "Dangerous" states, with greater than 99% probability of failure on the next step, are circled in red.

Figure 3 illustrates the distribution of mesh points for Case 1, using states with the greatest variance within the mesh ($\dot{q}_4$, $\dot{q}_5$, and $\dot{q}_1$) as three representative axes. Recall that

even without added perturbations, the control policy results in a chaotic set of reachable states; these points fall within the blue region. The red points show states visited due to perturbations and from which failures are nearly certain.

B. Case 2

In this scenario, we include push disturbances during the training, again using a reward function that primarily encourages forward motion, with a small penalty on energy use. Every 0.008 (s) interval during training, there is a 5 percent chance that the model will receive a push for the next 0.008 (s), with a 50/50 chance in either the forward or the back ward direction. We only introduce pushes at the COM of torso, and the magnitudes of the push are restricted to ± 1000 (N). As in Case 1, the policy with the maximum reward that we obtain for this new scenario also results in an asymmetric gait, now with a slower average gait cycle time of approximately 0.5 (s). We pick a threshold time of 0.45 (s) after which any post impact state (for left foot) with the ground is considered for meshing. To examine the effects of introducing disturbances during the training, we generate a mesh for the same disturbance profile (4 possible pushes, plus one no-push case) used in Case 1; for $d_{tr} = 0.6$, this leads to a mesh of 857 states. For the probability distribution given in Eq. 10, the MFPT for the system now comes out to about 10,467 steps, as compared with 32 steps for Case 1. An improvement in performance is an intuitively expected result, but our focus is on illustrating that our meshing tools allow us to quantify the improvement in robustness. We also generate the mesh for different d_{tr} values, shown in Table II.

d_{tr}	0.5	0.6	0.7
N (# of mesh points)	1705	857	574

TABLE II. N VERSUS d_{tr}, SHOWING $n \approx 3.25$ FOR CASE 2.

To analyze a more interesting disturbance profile and to check how the policy performs for magnitudes beyond the ones used for training, we generate a more complex mesh involving more varied disturbances, as shown in Fig. 4 As
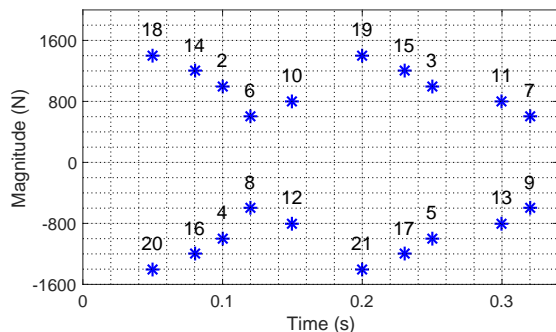

Fig. 4. Disturbance profile for 20 push types.

with the previous mesh, in addition to the 20 disturbances shown, we also include a no push scenario. This mesh has a total of 20,660 states which is significantly higher than the previous mesh but because we explored the mesh for more

disturbances, it also allows us to do much more interesting analyses. For example, if we set the probability distribution of the disturbances such that there is 0.6 probability (60% chance) of no push and 20% chance for disturbance 6 and disturbance 7 each, then we get a MFPT of infinity indicating that the walker will always recover under such disturbances. Similarly, if we set the probability disturbance such that probability of no push is 0.6, and probability of disturbance 8 and 9 is 0.2 each, then we get a MFPT of infinity as well. But, if we set the distribution such that probability of no push is 0.6 and the probability of disturbance 6, 7, 8 and 9 is 0.1 each, we get an MFPT of 32,260 showing that the *mixing effects* of disturbance 6, 7 and disturbance 8, 9 all combined over time will now create occasional failures, reducing the MFPT of the system significantly. Disturbances 6 and 7 correspond to pushes in the forward direction of magnitude 600 (N) and disturbance 8 and 9 correspond to the backward pushes of the same magnitude. We can also study the relative sensitivity of particular disturbances in the profile. For this we take an example distribution given by:

$$P(\gamma) = \begin{cases} 0.4, & \text{no disturbance} \\ 0.5, & \text{disturbance of interest} \\ 0.1/19, & \text{else.} \end{cases} \quad (12)$$

A plot showing the corresponding performance of the DRL policy for Case 2 is shown in Fig. 5. This is an interesting plot because it shows the coupled effect of direction and time of impact of the disturbance matters significantly. For example, we see that some disturbances of higher magnitude have a better MFPT than some disturbances of lower magnitude because they happen at different time instances.
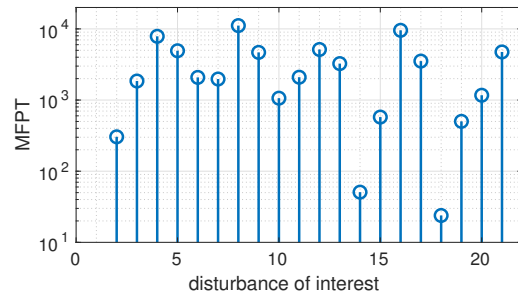

Fig. 5. MFPT variations as one disturbance of interest becomes more likely. The MFPT is shown on log scale to make the plot more readable.

Figures 6 and 7 show subsets of the full mesh corresponding to the reachable state space when each of two different subsets of the disturbance profiles shown in Fig. 4 are allowed. For Fig. 6, we exclude any pushes occurring before 0.1 seconds in the gait cycle; i.e., profiles 14, 16, 18 and 20 are excluded. For Fig. 7, all 20 push disturbances are included. For both figures, we assume a 2% chance of a push during each gait cycle, with the push type drawn with uniform probability from the allowable subset of pushes.

In Fig. 6, both subplots show the mesh points visited for this noise case, with darker points representing much more frequently-visited states. At right, magenta "+" symbols are

overlaid to show a set of 250 consecutive states visited (chaotically) when there is no noise during post-training testing of the policy. The MFPT predicted by the mesh is about 958,000 gait cycles. Once all 20 pushes are allowed (Fig. 7), this drops to a MFPT of only about 4,900 gait cycles. In this latter case, we can see that the system now visits a significant number of "unsafe" states (shown in red) that department significantly from the chaotic variability of locomotion when there is no noise.
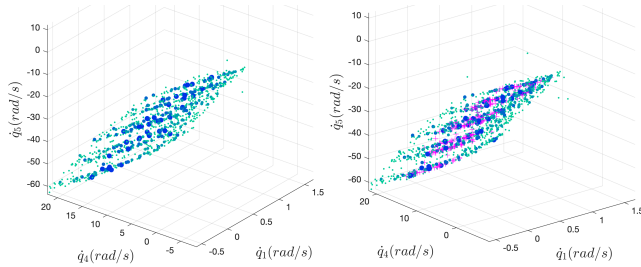

Fig. 6. A 3D slice of the full 13D Poincaré states generated to mesh the policy trained in Case 2, when all but disturbance profiles 14, 16, 18 and 20 are included, i.e., excluding pushes occurring before 0.1 seconds.
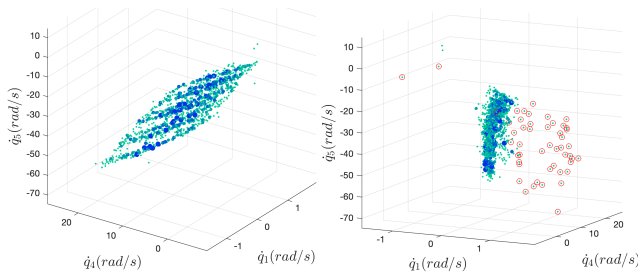

Fig. 7. A 3D slice of the full 13D Poincaré states generated to mesh the policy trained in Case 2, when all 20 timing and magnitude combinations shown in Fig. 4 are included. As in Fig. 3, the subplot at right highlights states from which immediate failure has probability greater that 99%.

V. CONCLUSIONS

We have demonstrated the effectiveness of our meshing tools in analyses of the polices created via deep reinforcement learning. To quantify the robustness of these policies, we create meshes for different disturbance combinations that can happen at various times in the gait cycle and then set transitions among mesh points based on particular assumed probability distributions of the disturbances. The policy trained for only forward motion is significantly less robust to perturbations as compared to the policy trained with impacts, which is intuitive but our tools gave us an estimate on the actual performance improvement. Such an estimate is important if we want to improve the performance of training by changing various parameters that can affect the outcome of the trained policy. In addition, by performing the analyses on various probability distributions we show that our tools also allow us to plot performance trends which can help us to understand the effect each individual disturbance in the disturbance profile has on the overall performance of the policy. We also show how the mixing effects of these disturbances can have significant effect on robustness.

In showing the applicability of our meshing tools to analyze the policies obtained via deep reinforcement learning, we have also created an indirect feedback loop that can be used to improve the performance of the policies by tuning various elements of the training framework. A future goal is to integrate this feedback loop within the training framework. We also plan to explore how increasing the number of discrete perturbation types affects mesh size. Although we consider toy sets of perturbations here, for illustrative purposes, practical use of these tools should cope with denser sets of perturbations. Finally, another important goal is to explore how various reward functions and/or disturbances during training may increase the degree to which a DRL policy contracts the dynamics; reducing the dimensionality of the state space visited would increase the practicality of meshing. We show some initial success in this by reducing dimensionality from $n = 4.23$ to $n = 3.25$ when training includes pushes. Future work is planned to investigate if this a repeatable trend and whether more extreme perturbations during training can further improve contraction.

.

REFERENCES

[1] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Auton. Robots*, vol. 40, no. 3, pp. 429–455, 2016.

[2] S. Feng, E. C. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization-based full body control for the darpa robotics challenge," *J. Field Robotics*, vol. 32, pp. 293–312, 2015.

[3] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," 2017.

[4] M. Posa, S. Kuindersma, and R. Tedrake, "Optimization and stabilization of trajectories for constrained dynamical systems," in *Proc. IEEE Int. Conf. on Robotics and Autom. (ICRA)*, May 2016, pp. 1366–1373.

[5] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics," in *Proc. IEEE Int. Conf. on Robotics and Autom. (ICRA)*, 2016, pp. 1447–1454.

[6] S. Sovero, N. Talele, C. Smith, N. Cox, T. Swift, and K. Byl, "Initial data and theory for a high specific-power ankle exoskeleton device," in *2016 Int. Symp. on Exper. Robotics (ISER)*, D. Kulić, Y. Nakamura, O. Khatib, and G. Venture, Eds. Springer, 2017, pp. 355–364.

[7] G. Bellegarda, N. Talele, and K. Byl, "Nonintuitive optima for dynamic locomotion: The Acrollbot," 05 2018, pp. 3130–3136.

[8] K. Byl and R. Tedrake, "Metastable walking machines," *I. J. Robotics Res.*, vol. 28, pp. 1040–1064, 2009.

[9] C. O. Saglam and K. Byl, "Robust policies via meshing for metastable rough terrain walking," in *Proc. Robotics: Science and Systems*, 2014.

[10] ——, "Meshing hybrid zero dynamics for rough terrain walking," in *IEEE Int. Conf. on Robotics and Autom. (ICRA)*, 2015, pp. 5718–5725.

[11] N. Talele and K. Byl, "Mesh-based methods for quantifying and improving robustness of a planar biped model to random push disturbances," in *Proc. American Control Conf. (ACC)*, 2019 (in press).

[12] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 5026–5033.

[13] N. Talele and K. Byl, "Methods and performance analyses for design and feedback control of efficient and robust planar biped walking," in *Proc. American Control Conf. (ACC)*, 2019 (in press).

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[15] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "OpenAI baselines," https://github.com/openai/baselines, 2017.

[16] C. O. Saglam, "Tractable Quantication of Metastability for Robust Bipedal Locomotion," PhD thesis, UCSB, 2015.